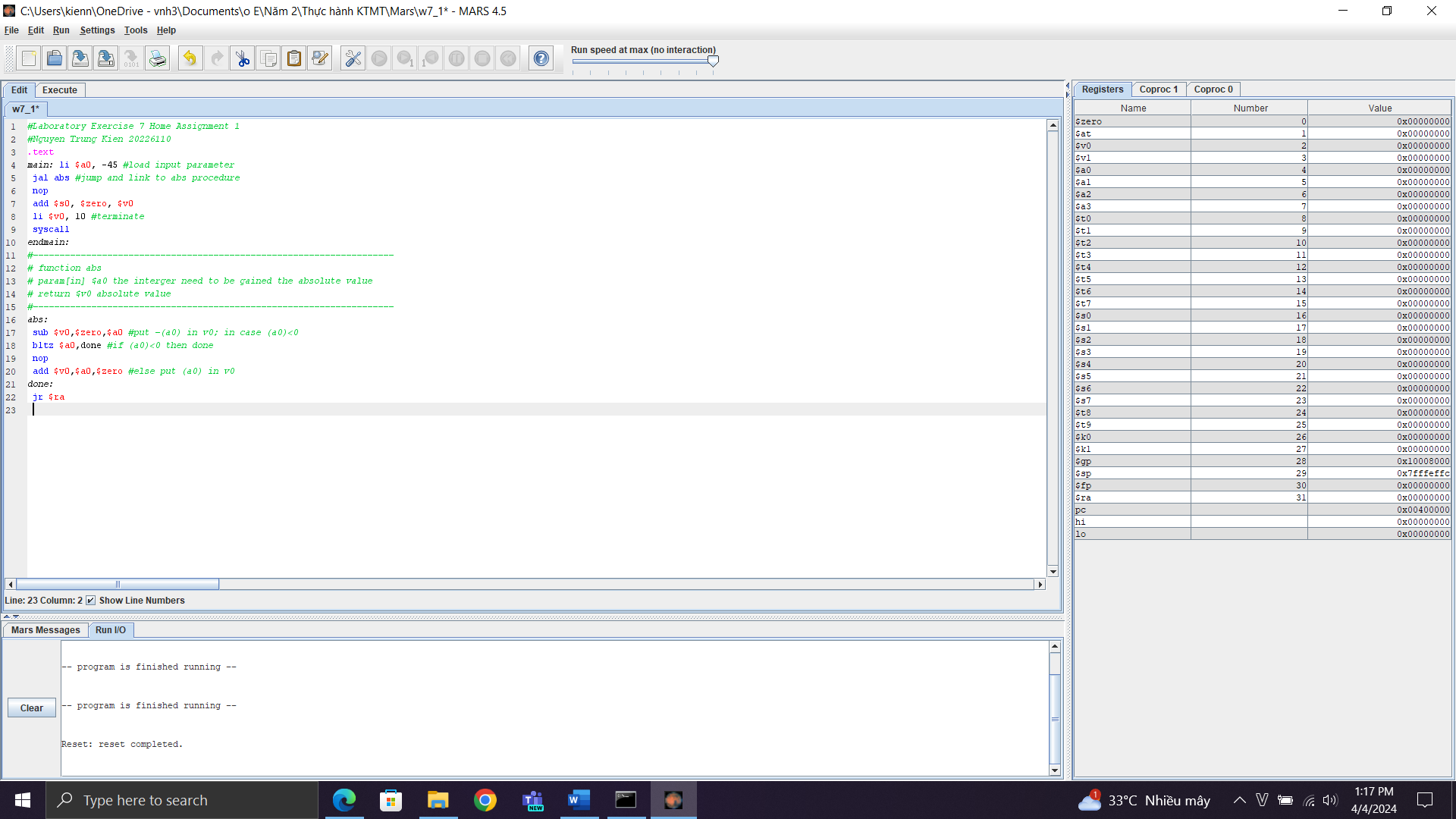
**Báo cáo Thực hành Kiến trúc máy tính tuần 7**

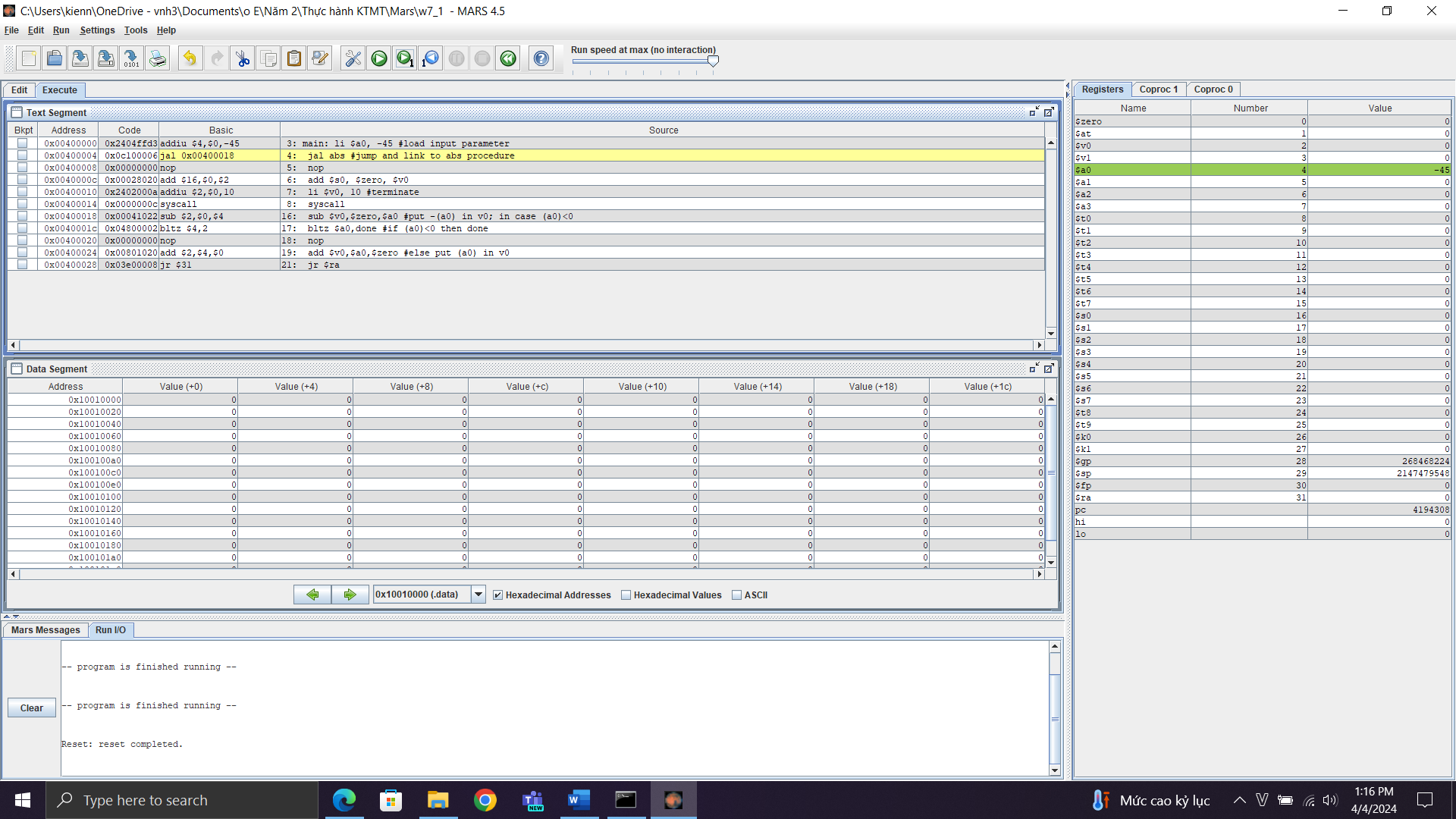
**Họ và tên: Nguyễn Trung Kiên**

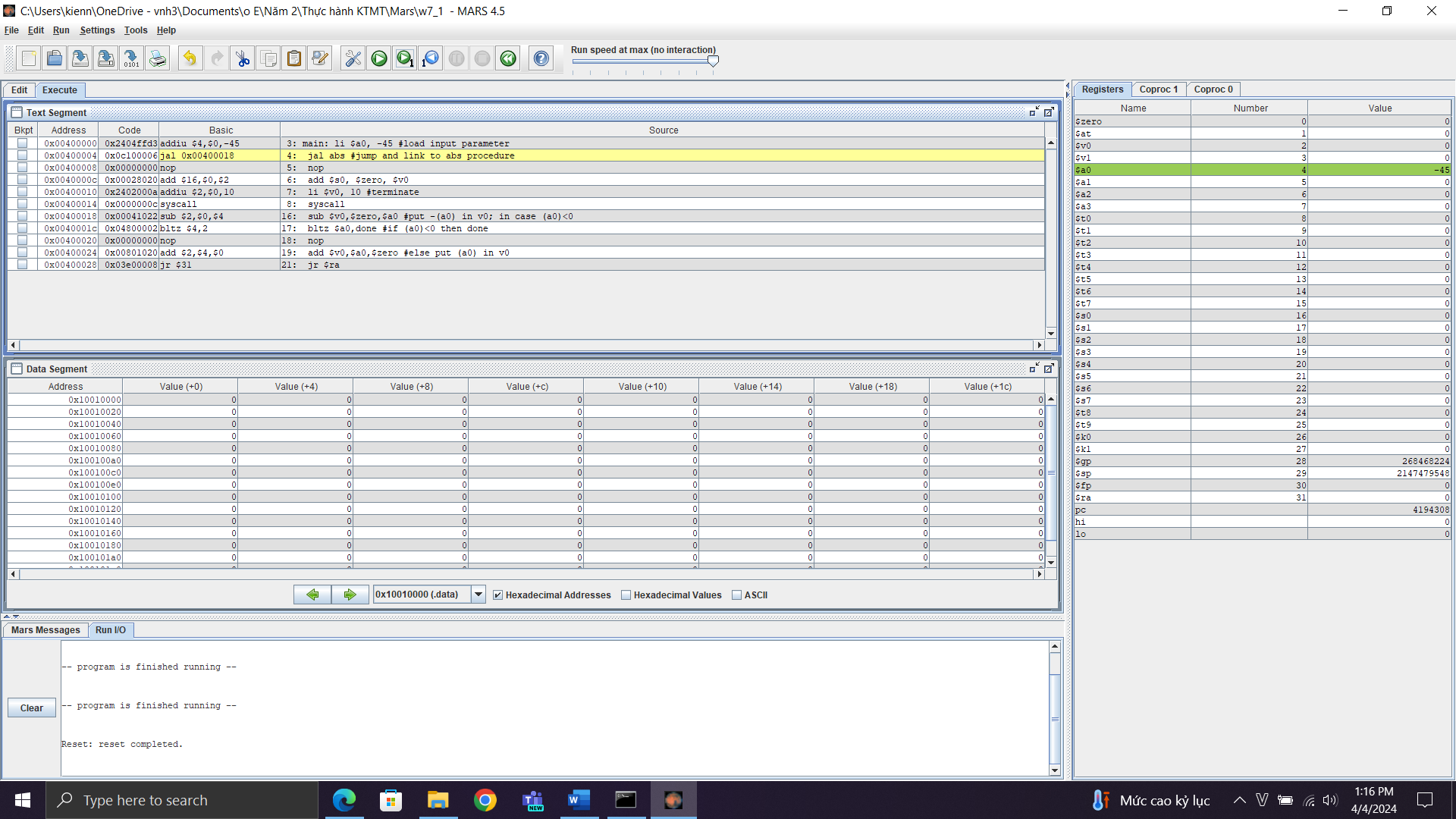
**MSSV:20226110**

Assignment 1



* Trước khi chạy lệnh “jal abs”:





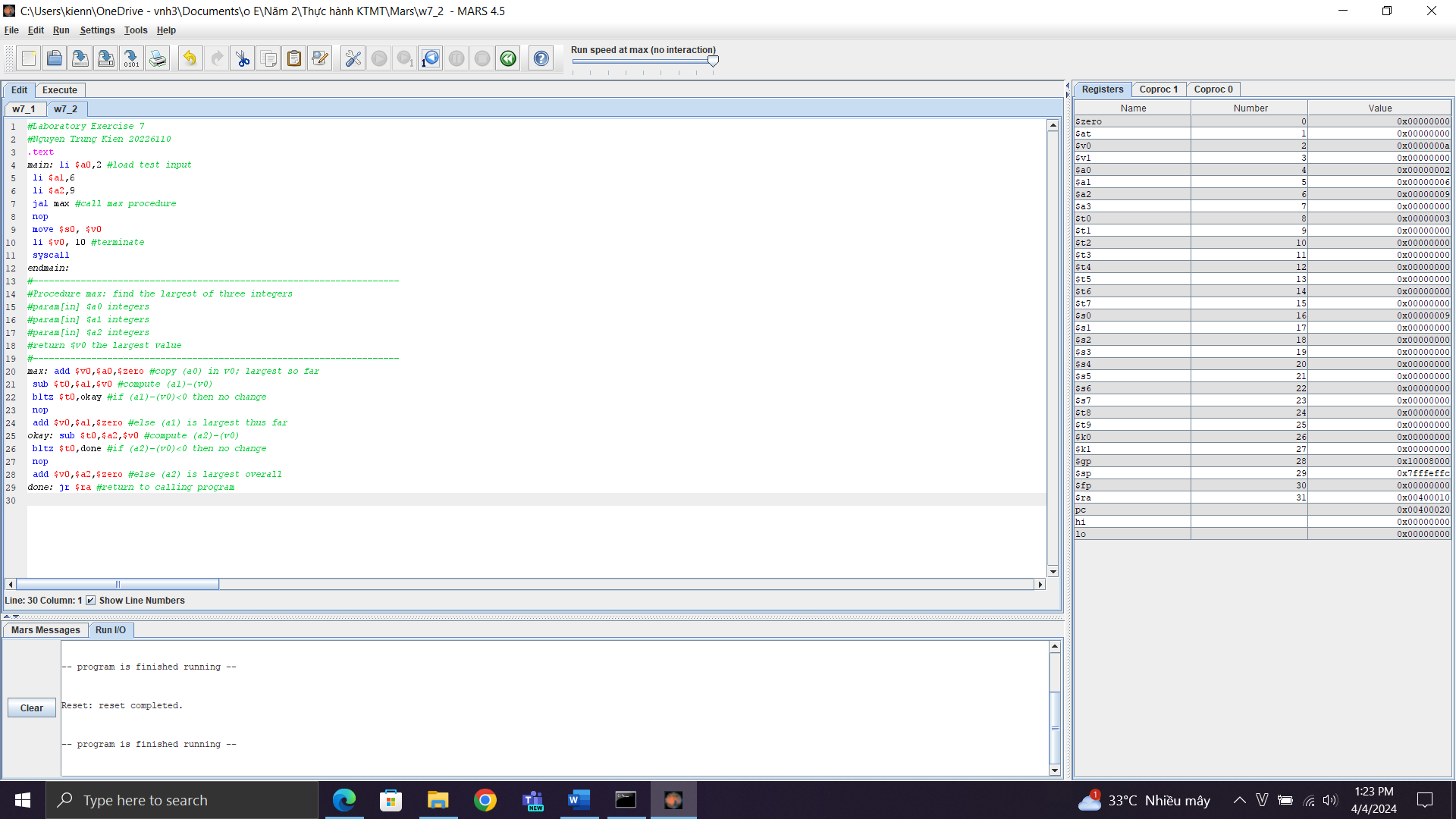
* Sau khi chạy lệnh “jal abs”:



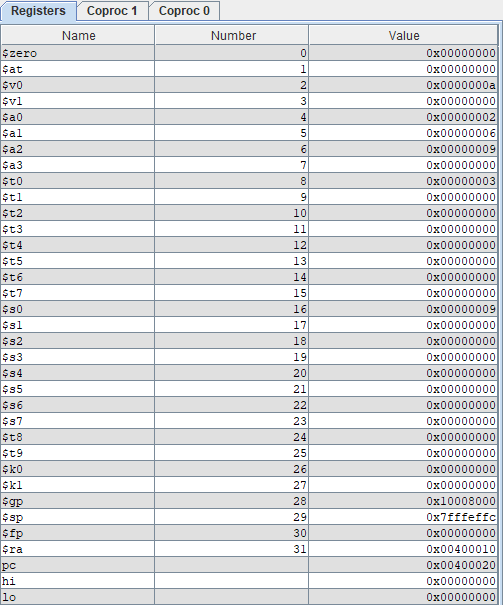


* Khi chạy lệnh “jal abs” (địa chỉ lệnh 0x00400004), lệnh tiếp theo của lệnh đó được lưu vào thanh ghi $ra (0x00400008). Đồng thời, thanh ghi pc được gán giá trị 0x00400018 (địa của của nhãn abs)

Assignment 2

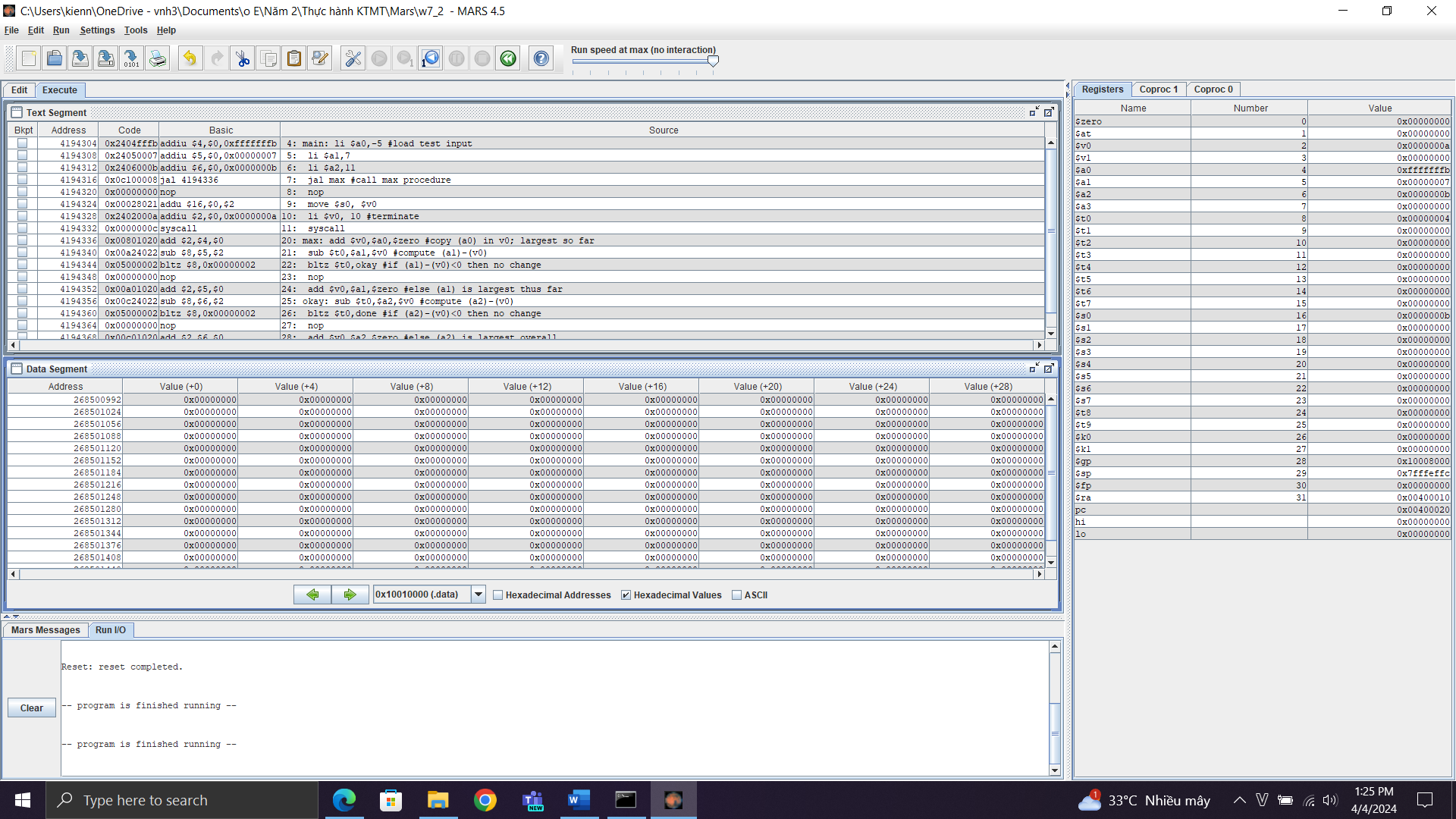


* Chạy test mẫu của đề bài:



Thanh ghi $s0 = 9 là kết quả của chương trình => Đúng với kỳ vọng.

* Chạy test tự tạo: $a0 = -5, $a1 = 7, $a2 = 11



Thanh ghi $s0 = 0x0000000b hay là 11 là kết quả của chương trình => Đúng với lý thuyết.

* Khi chạy lệnh “jal abs”, lệnh tiếp theo của lệnh đó được lưu vào thanh ghi $ra. Đồng thời, thanh ghi pc được gán giá trị địa của của nhãn max. Sau khi chạy đến nhãn done, lệnh “jr $ra” gán thanh ghi pc giá trị địa chỉ của thanh ghi $ra

Assignment 3

#Laboratory Exercise 7, Home Assignment 3

.text

li $s0, 6

li $s1, 9

push:

addi $sp,$sp,-8 #adjust the stack pointer

sw $s0,4($sp) #push $s0 to stack

sw $s1,0($sp) #push $s1 to stack

work:

nop

nop

nop

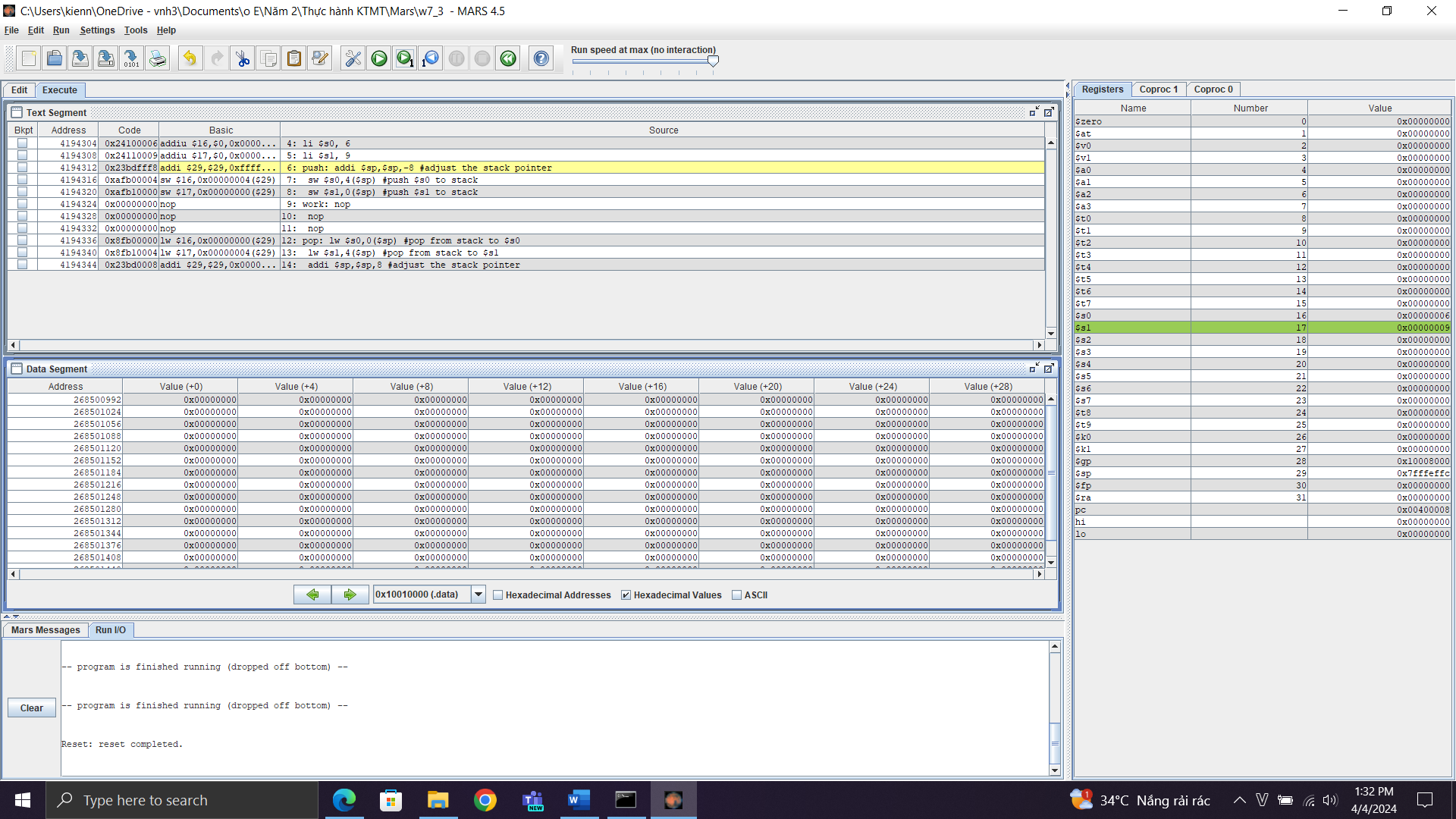
pop:

lw $s0,0($sp) #pop from stack to $s0

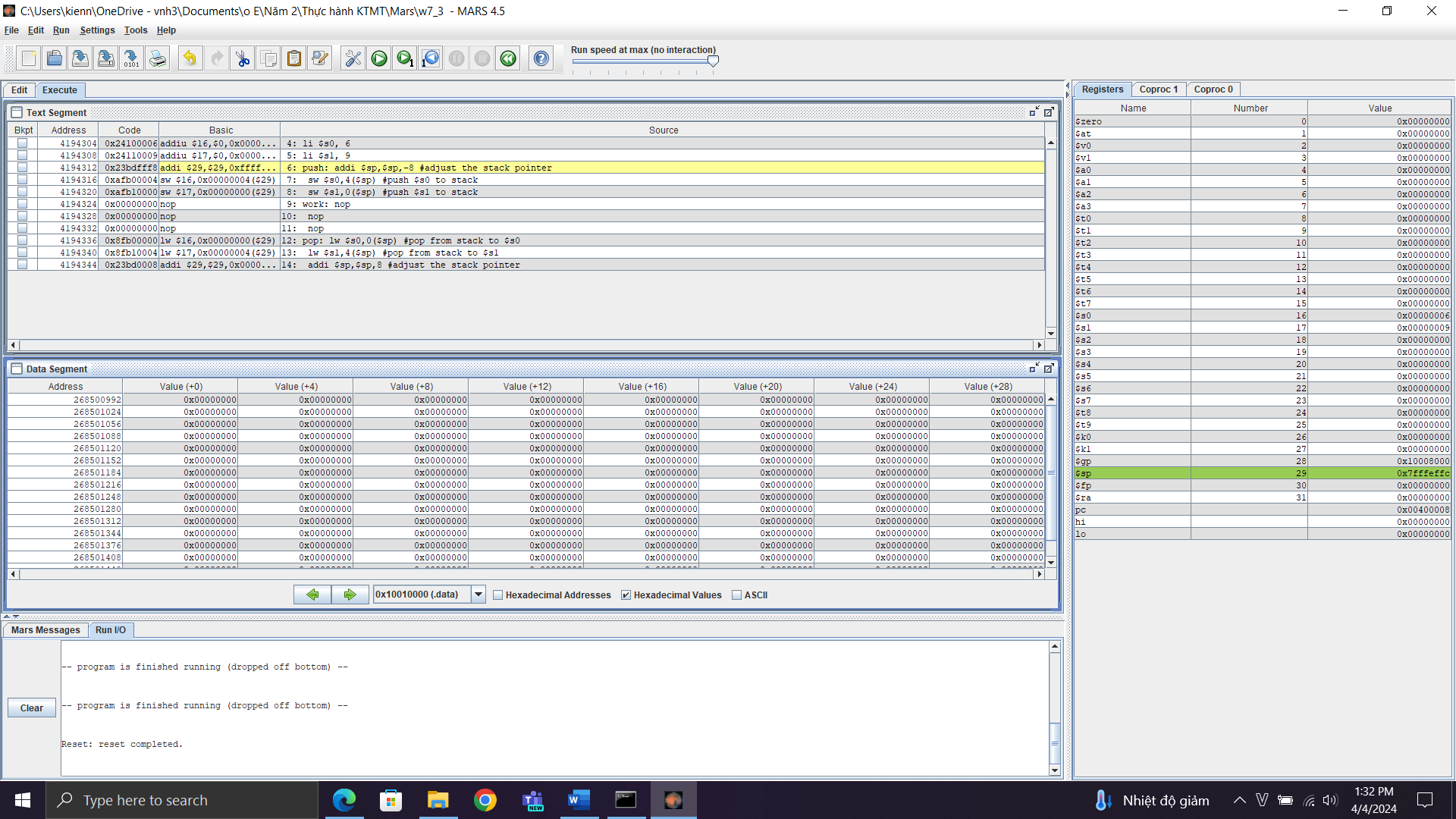
lw $s1,4($sp) #pop from stack to $s1

addi $sp,$sp,8 #adjust the stack pointer

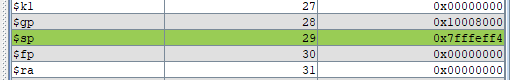
* Gán thanh ghi $s0 với giá trị 6 và gán thanh ghi $s1 với giá trị 9



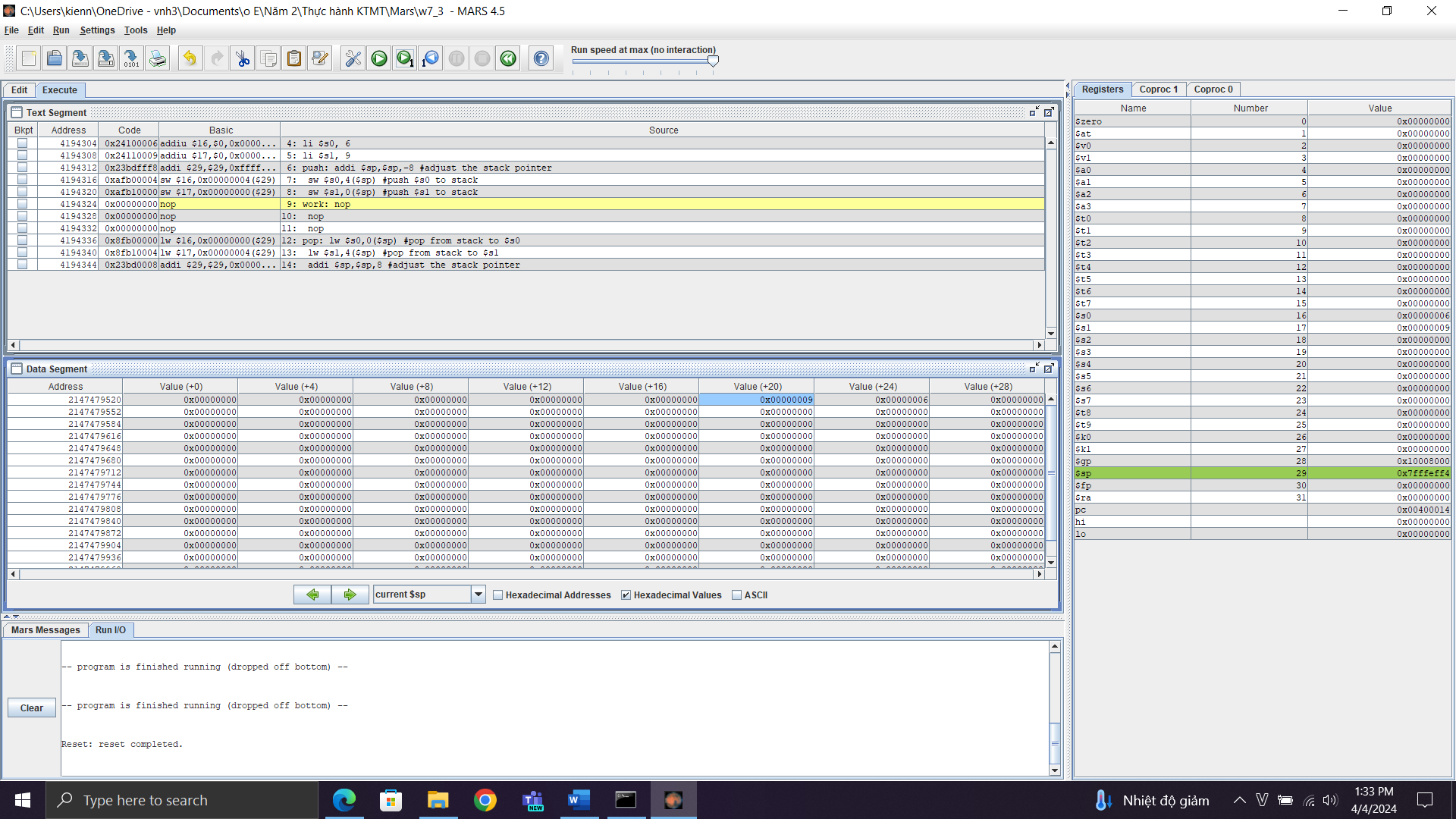
* Lệnh addi $sp,$sp,-8 ở nhãn “push” là để giảm giá trị địa chỉ của thanh ghi $sp đi 8 đơn vị. Tức là có sự cấp phát bộ nhớ 8 byte trong stack.
* Địa chỉ ban đầu của $sp



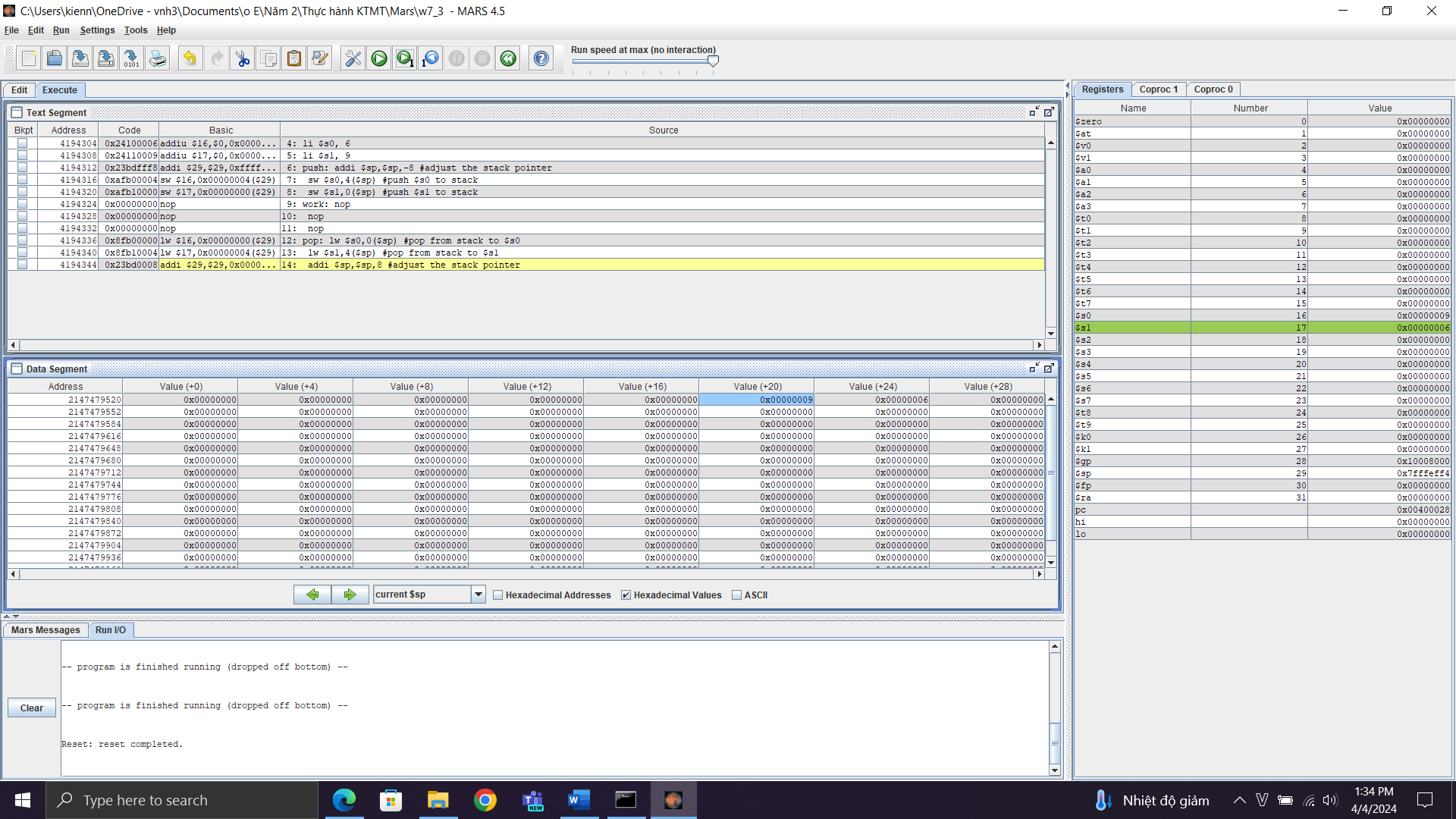
* Địa chỉ $sp sau khi trừ đi 8 đơn vị



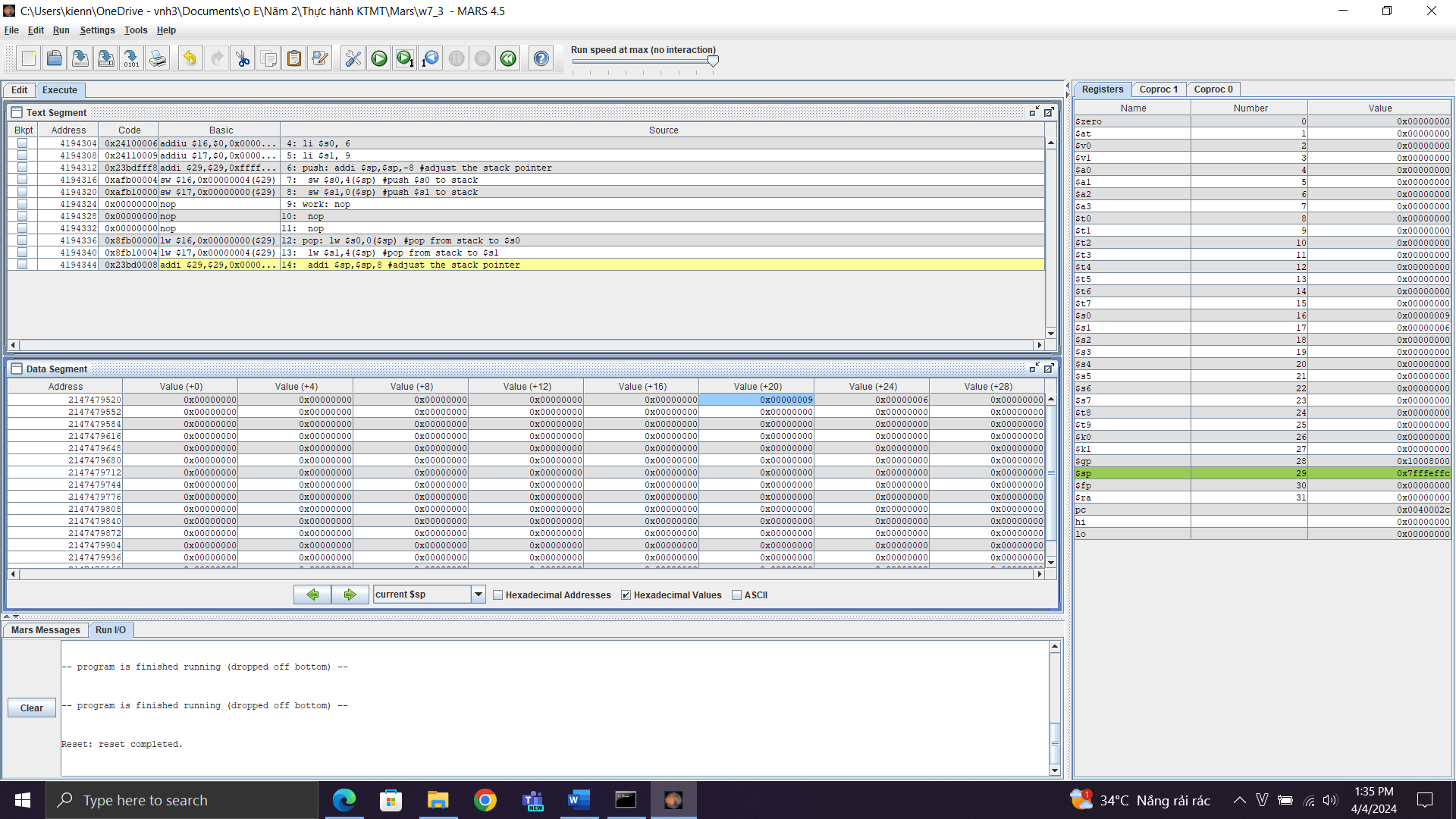
* Sau đó ta lưu giá trị của $s0 vào $sp + 4 và $s1 vào $sp + 0



* Trong nhãn “pop”, 2 lệnh lw thực hiện lưu giá trị có địa chỉ $sp + 0 vào $s0 và $sp + 4 vào $s1. Điều này làm giá trị 2 thanh ghi đổi chỗ cho nhau



* Lệnh addi $sp,$sp,8 trả lại giá trị địa chỉ của $sp về ban đầu



Assignment 4

#Laboratory Exercise 7, Home Assignment 4

.data

Message: .asciiz "Ket qua tinh giai thua la: "

.text

main:

jal WARP

print:

add $a1, $v0, $zero # $a0 = result from N!

li $v0, 56

la $a0, Message

syscall

quit: li $v0, 10 #terminate

syscall

endmain:

#----------------------------------------------------------------------

#Procedure WARP: assign value and call FACT

#----------------------------------------------------------------------

WARP:

sw $fp,-4($sp) #save frame pointer (1)

addi $fp,$sp,0 #new frame pointer point to the top (2)

addi $sp,$sp,-8 #adjust stack pointer (3)

sw $ra,0($sp) #save return address (4)

li $a0,6 #load test input N

jal FACT #call fact procedure

nop

lw $ra,0($sp) #restore return address (5)

addi $sp,$fp,0 #return stack pointer (6)

lw $fp,-4($sp) #return frame pointer (7)

jr $ra

wrap\_end:

#----------------------------------------------------------------------

#Procedure FACT: compute N!

#param[in] $a0 integer N

#return $v0 the largest value

#----------------------------------------------------------------------

FACT:

sw $fp,-4($sp) #save frame pointer

addi $fp,$sp,0 #new frame pointer point to stack’s top

addi $sp,$sp,-12 #allocate space for $fp,$ra,$a0 in stack

sw $ra,4($sp) #save return address

sw $a0,0($sp) #save $a0 register

slti $t0,$a0,2 #if input argument N < 2

beq $t0,$zero,recursive#if it is false ((a0 = N) >=2)

nop

li $v0,1 #return the result N!=1

j done

nop

recursive:

addi $a0,$a0,-1 #adjust input argument

jal FACT #recursive call

nop

lw $v1,0($sp) #load a0

mult $v1,$v0 #compute the result

mflo $v0

done: lw $ra,4($sp) #restore return address

lw $a0,0($sp) #restore a0

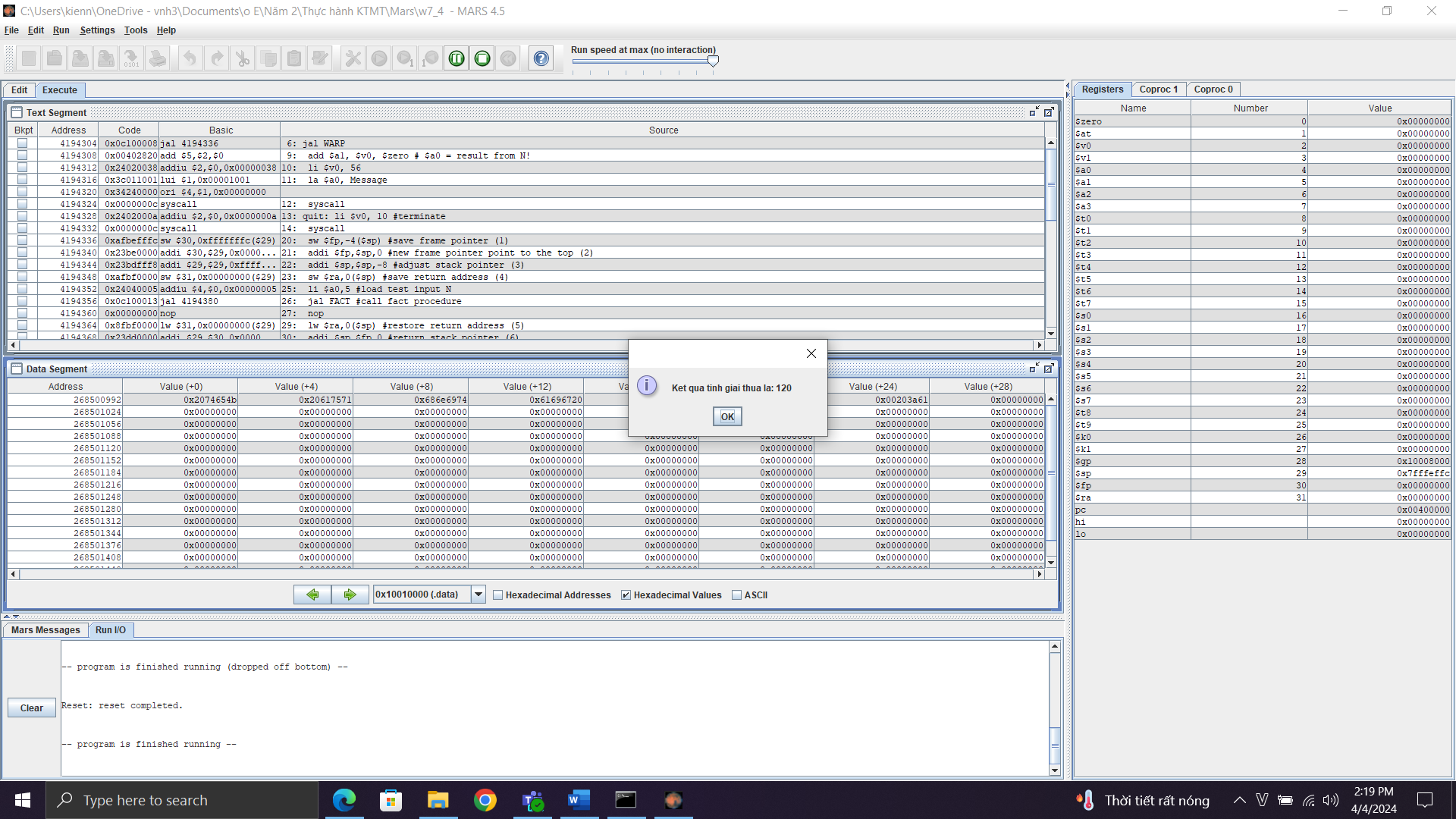
addi $sp,$fp,0 #restore stack pointer

lw $fp,-4($sp) #restore frame pointer

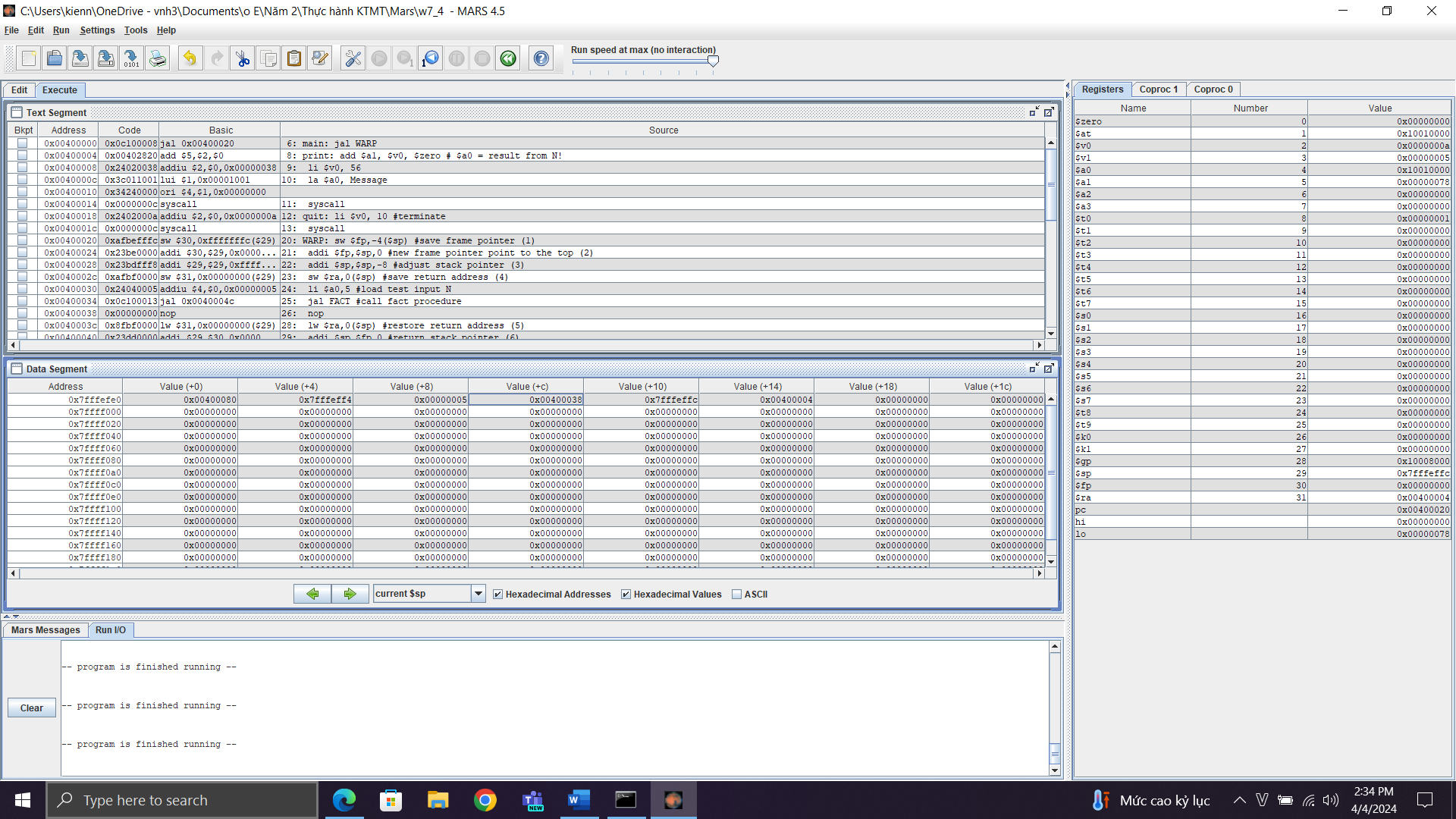
jr $ra #jump to calling

fact\_end:

* Với n = 5, $a0 = 5, kết quả chương trình là 120



* Chương trình chạy đúng kết quả mong đợi
* Sự thanh đổi của thanh ghi $sp trong chương trình



* Bảng biểu diễn giá trị các ngăn nhớ trong trường hợp n = 3 (Tính 3!)

|  |  |
| --- | --- |
| 0x 7fffef8 | $fp = 0x00000000 |
| 0x7fffeff4 | $ra = 0x00400004 |
| 0x7fffeff0 | $fp = 0x7fffeffc |
| 0x7fffefec | $ra = 0x00400038 |
| 0x7fffefe8 | $a0 = 0x00000003 |
| 0x7fffefe4 | $fp = 0x7fffeff4 |
| 0x7fffefe0 | $ra = 0x00400080 |
| 0x7fffefdc | $a0 = 0x00000002 |
| 0x7fffefd8 | $fp = 0x7fffefe4 |
| 0x7fffefd4 | $ra = 0x00400080 |
| 0x7fffefd0 | $a0 = 0x00000001 |

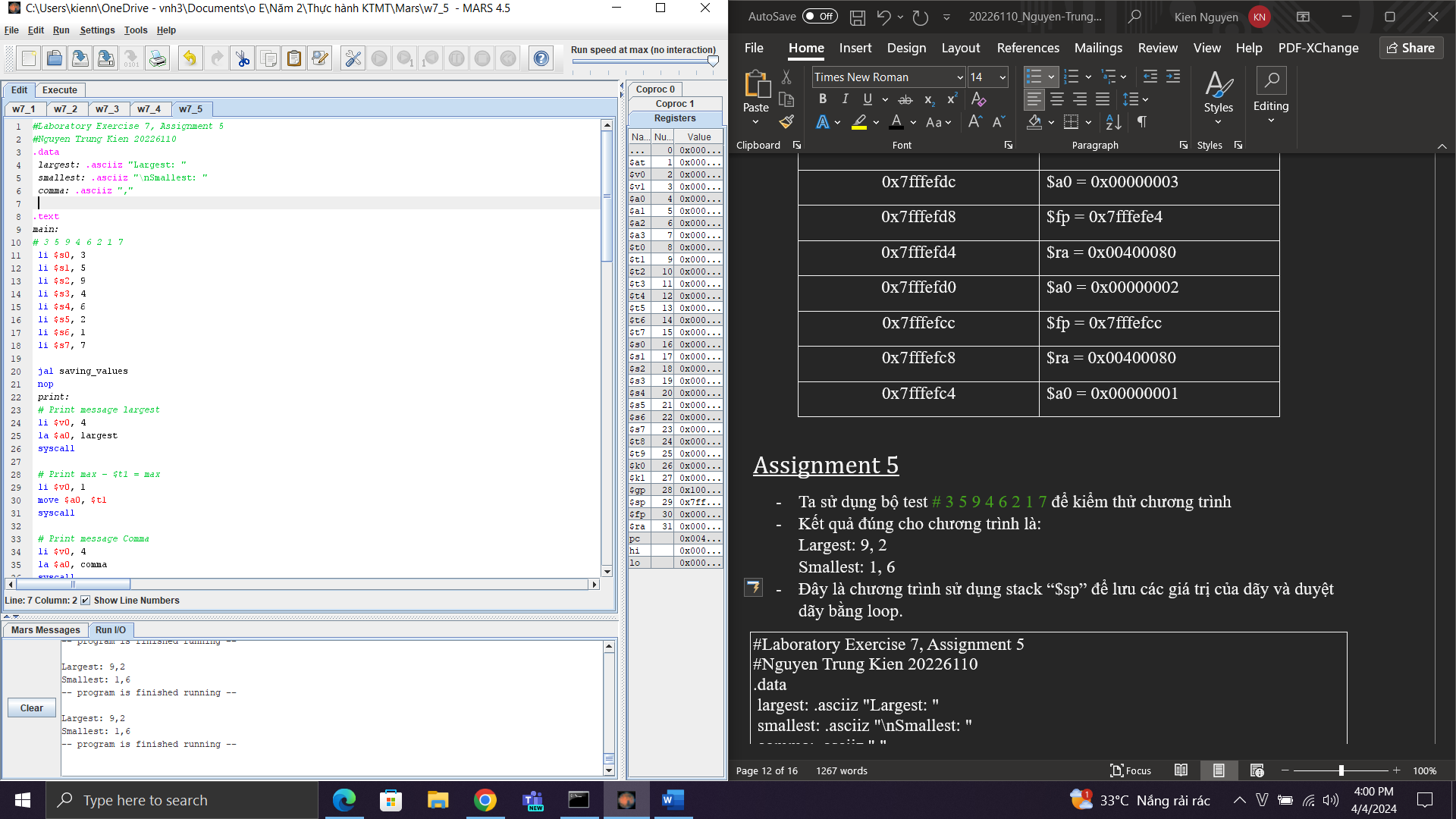
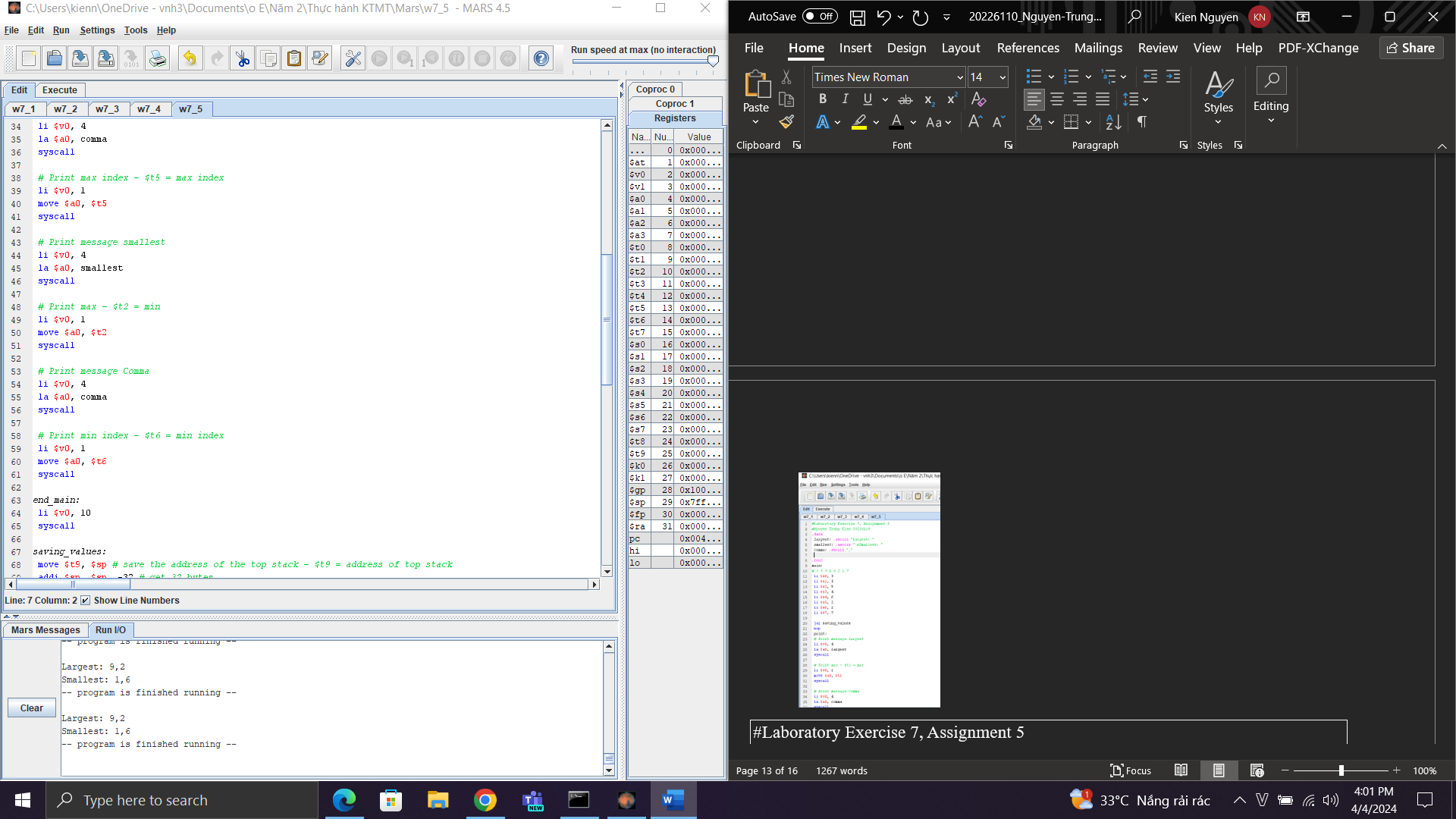
Assignment 5

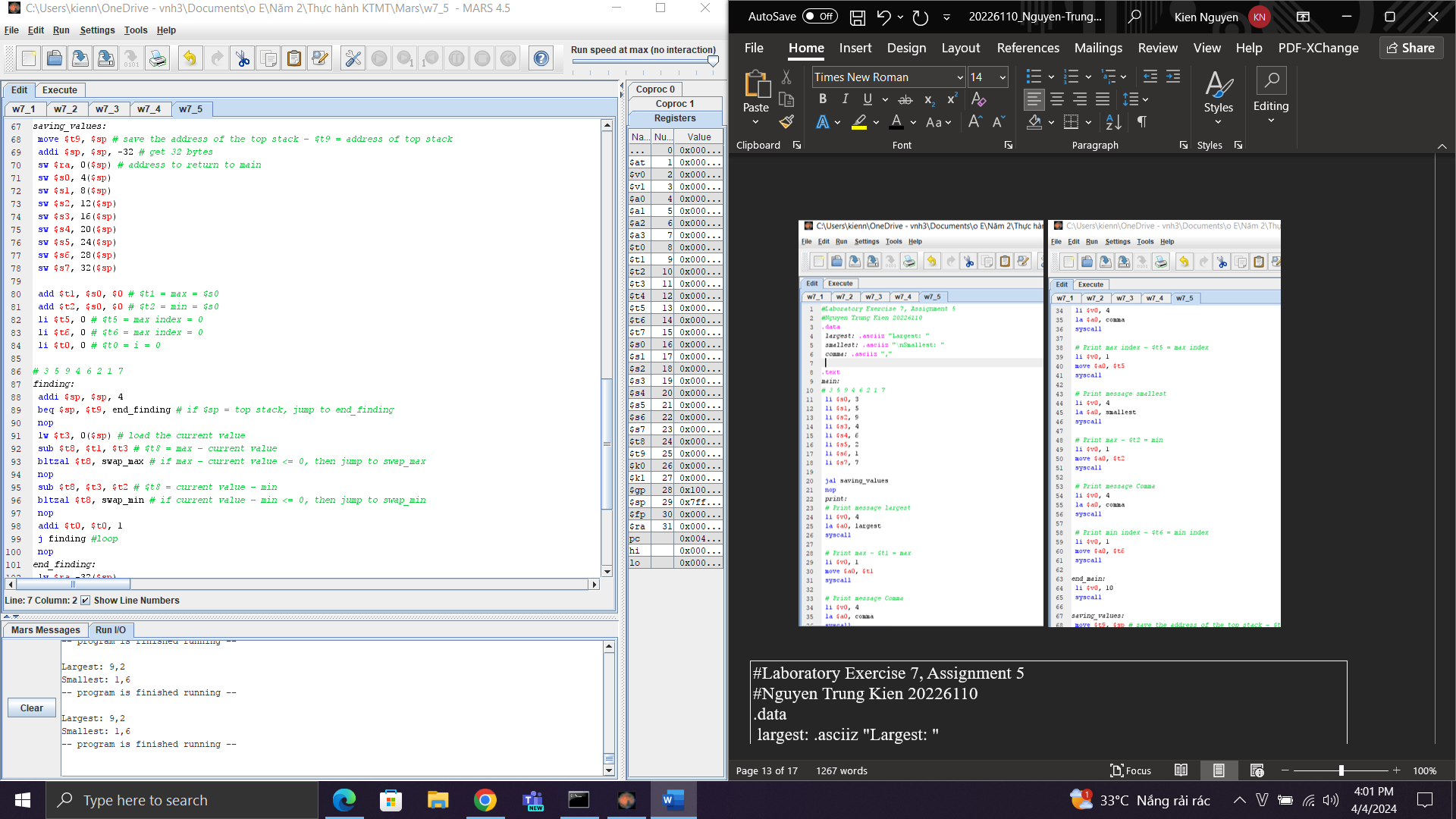
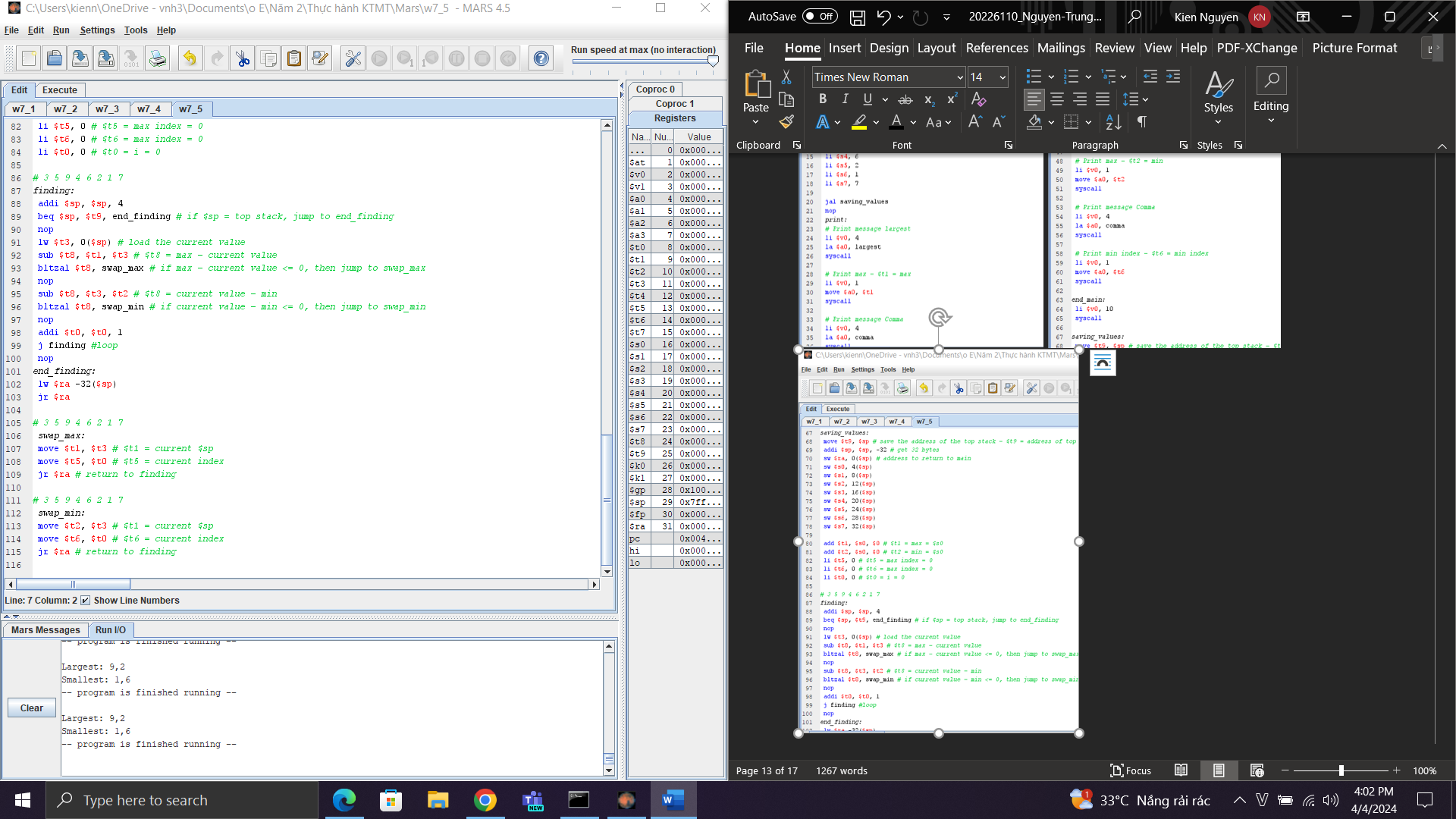
* Ta sử dụng bộ test # 3 5 9 4 6 2 1 7 để kiểm thử chương trình
* Kết quả đúng của test case này cho chương trình là:

Largest: 9, 2

Smallest: 1, 6

* Đây là chương trình sử dụng stack “$sp” để lưu các giá trị của dãy và duyệt dãy bằng cách gọi lại hàm tìm max min và cập nhật vị trí của $sp.
* Hàm “finding” tìm max min bằng cách so sánh và cập nhật 2 giá trị max min, sau đó cập nhật index của 2 giá trị đó và lưu lại vào thanh ghi.

#Laboratory Exercise 7, Assignment 5

#Nguyen Trung Kien 20226110

.data

largest: .asciiz "Largest: "

smallest: .asciiz "\nSmallest: "

comma: .asciiz ","

.text

main:

# 3 5 9 4 6 2 1 7

li $s0, 3

li $s1, 5

li $s2, 9

li $s3, 4

li $s4, 6

li $s5, 2

li $s6, 1

li $s7, 7

jal saving\_values

nop

print:

# Print message largest

li $v0, 4

la $a0, largest

syscall

# Print max - $t1 = max

li $v0, 1

move $a0, $t1

syscall

# Print message Comma

li $v0, 4

la $a0, comma

syscall

# Print max index - $t5 = max index

li $v0, 1

move $a0, $t5

syscall

# Print message smallest

li $v0, 4

la $a0, smallest

syscall

# Print max - $t2 = min

li $v0, 1

move $a0, $t2

syscall

# Print message Comma

li $v0, 4

la $a0, comma

syscall

# Print min index - $t6 = min index

li $v0, 1

move $a0, $t6

syscall

end\_main:

li $v0, 10

syscall

saving\_values:

move $t9, $sp # save the address of the top stack - $t9 = address of top stack

addi $sp, $sp, -32 # get 32 bytes

sw $ra, 0($sp) # address to return to main

sw $s0, 4($sp)

sw $s1, 8($sp)

sw $s2, 12($sp)

sw $s3, 16($sp)

sw $s4, 20($sp)

sw $s5, 24($sp)

sw $s6, 28($sp)

sw $s7, 32($sp)

add $t1, $s0, $0 # $t1 = max = $s0

add $t2, $s0, $0 # $t2 = min = $s0

li $t5, 0 # $t5 = max index = 0

li $t6, 0 # $t6 = max index = 0

li $t0, 0 # $t0 = i = 0

# 3 5 9 4 6 2 1 7

finding:

addi $sp, $sp, 4

beq $sp, $t9, end\_finding # if $sp = top stack, jump to end\_finding

nop

lw $t3, 0($sp) # load the current value

sub $t8, $t1, $t3 # $t8 = max - current value

bltzal $t8, swap\_max # if max - current value <= 0, then jump to swap\_max

nop

sub $t8, $t3, $t2 # $t8 = current value - min

bltzal $t8, swap\_min # if current value - min <= 0, then jump to swap\_min

nop

addi $t0, $t0, 1

j finding #loop

nop

end\_finding:

lw $ra -32($sp)

jr $ra

# 3 5 9 4 6 2 1 7

swap\_max:

move $t1, $t3 # $t1 = current $sp

move $t5, $t0 # $t5 = current index

jr $ra # return to finding

# 3 5 9 4 6 2 1 7

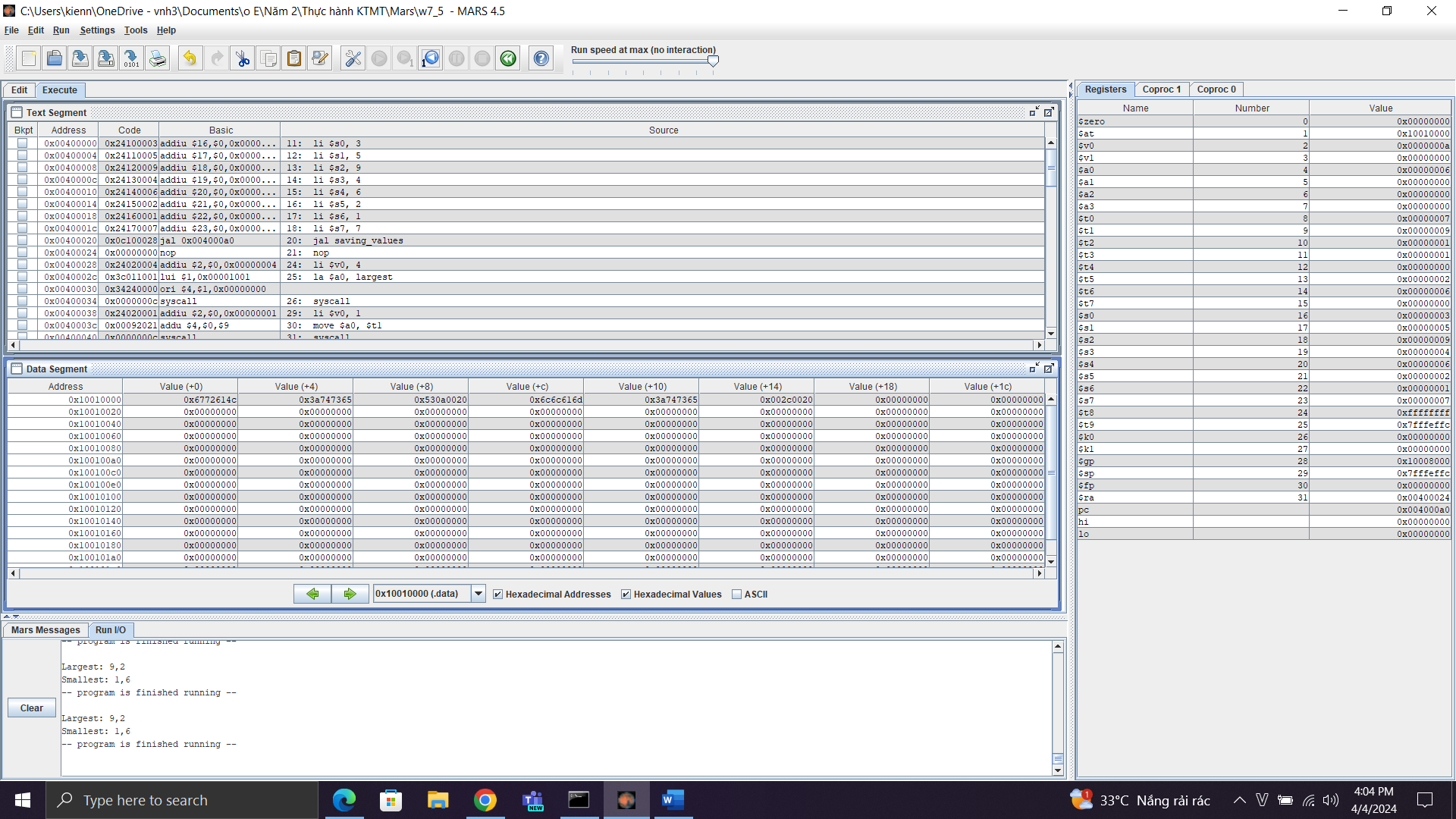
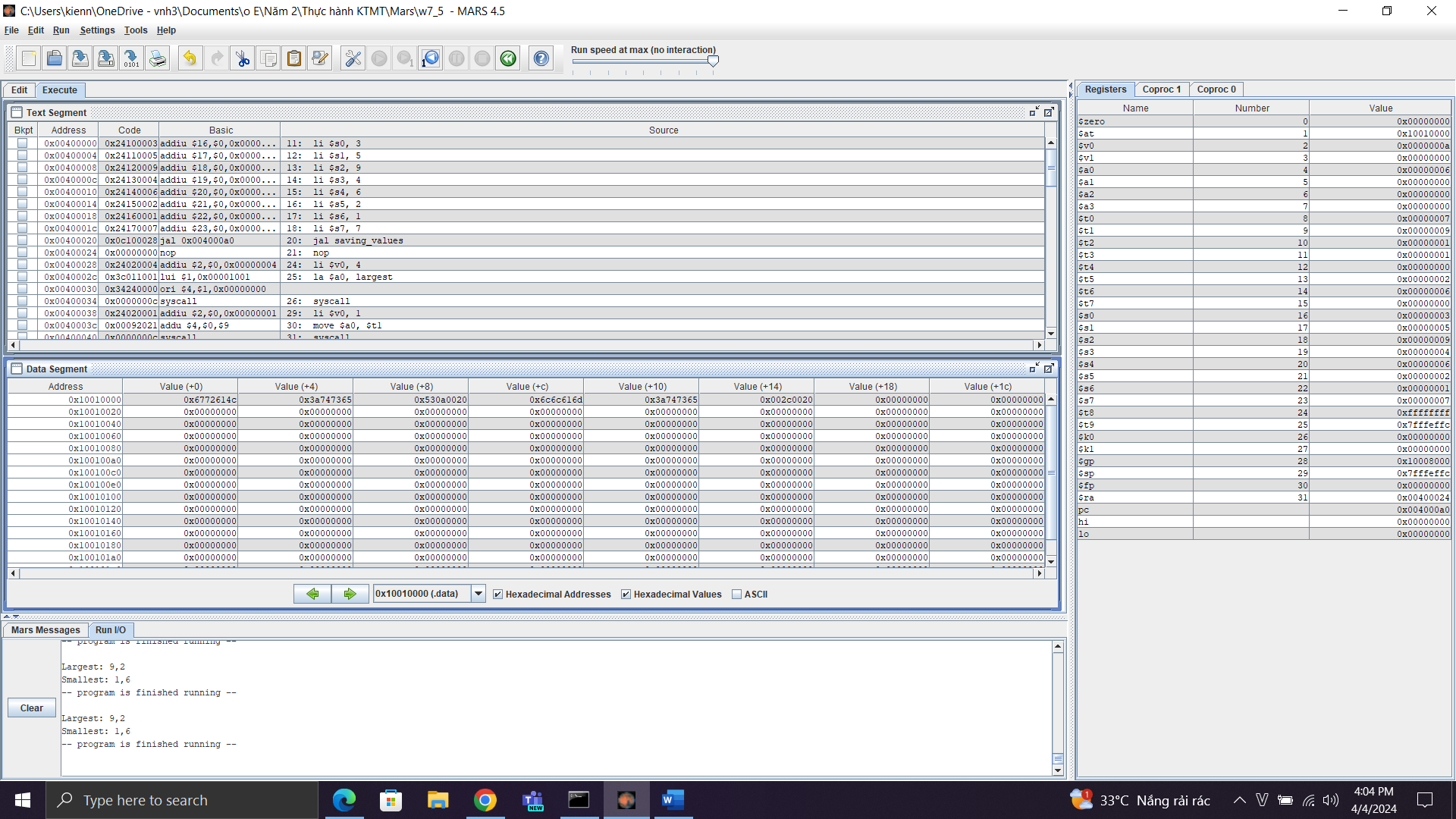
swap\_min:

move $t2, $t3 # $t1 = current $sp

move $t6, $t0 # $t6 = current index

jr $ra # return to finding

* Test: $s0 = 3, $s1 = 5, $s2 = 9, $s3 = 4, $s4 = 6, $s5 = 2, $s6 = 1, $s7 = 7
* Kết quả của các thanh ghi và bộ nhớ stack sau khi thực hiện chương trình trên:

* Kết quả này phù hợp và chính xác.